Nearest neighbor methods for regression & classification ...cont’d

3 methods we discussed...

- Nearest neighbor regression
- Weighted nearest neighbor regression
- Kernel regression
1-NN regression

Dataset of ($, $) pairs: $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$

Query point: $x_q$

1. Find “closest” $x_i$ in dataset

2. Predict

$$\hat{y}_q = y_{NN}$$

$K = 1$ Nearest Neighbors Kernel

Fit looks good for data dense in $x$ and low noise
Sensitive to regions with little data

Nearest Neighbors Kernel (K = 1)

Not great at interpolating over large regions…

Also sensitive to noise in data

Nearest Neighbors Kernel (K = 1)

Fits can look quite wild… Overfitting?
k-NN regression

Dataset of (\( x, y \)) pairs: \((x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\)
Query point: \(x_q\)

1. Find \(k\) closest \(x_i\) in dataset
   \(\{(x_{NN,1}, \ldots, x_{NN,k})\}\) such that for any \(x_i\) not in nearest \(k\) neighbors, \(\text{distance}(x_i, x_q) \geq \text{distance}(x_{NN,k}, x_q)\)

2. Predict
   \[
   \hat{y}_q = \frac{1}{k} \sum_{j=1}^{k} y_{NN,j}
   \]

k-NN in practice
k-NN in practice

Much more reasonable fit in the presence of noise

Boundary & sparse region issues

k-NN in practice

Discontinuities! Neighbor either in or out
3 methods we discussed...

- Nearest neighbor regression

- Weighted nearest neighbor regression

- Kernel regression

Weighted k-NN

Weigh more similar houses more than those less similar in list of k-NN

Predict:

\[ \hat{y}_q = \frac{c_{qNN1}y_{NN1} + c_{qNN2}y_{NN2} + c_{qNN3}y_{NN3} + \ldots + c_{qNNk}y_{NNk}}{\sum_{j=1}^{k} c_{qNNj}} \]
Kernel weights for \( d \geq 1 \)

Define: \( c_{qNNj} = \text{Kernel}_\lambda(\text{distance}(x_{NNj}, x_q)) \)

3 methods we discussed...

- Nearest neighbor regression
- Weighted nearest neighbor regression
- Kernel regression
Kernel regression

Instead of just weighting NN, weight all points

Predict:

\[ \hat{y}_q = \frac{\sum_{i=1}^{N} c_q y_i}{\sum_{i=1}^{N} c_q} = \frac{\sum_{i=1}^{N} \text{Kernel}_\lambda (\text{distance}(x_i, x_q)) * y_i}{\sum_{i=1}^{N} \text{Kernel}_\lambda (\text{distance}(x_i, x_q))} \]

Nadaraya-Watson kernel weighted average

Kernel regression in practice

Kernel has bounded support...
Only subset of data needed to compute local fit
Formalizing the idea of local fits

Contrasting with global average

A globally constant fit weights all points equally.

\[ \hat{y}_q = \frac{1}{N} \sum_{i=1}^{N} y_i = \frac{\sum_{i=1}^{N} c y_i}{\sum_{i=1}^{N} c} \]

**Boxcar Kernel (\(\lambda = 1\))**

parametric model: \(f(x) = w_0\)

\(\uparrow\)

**constant (intercept)**
Contrasting with global average

Kernel regression leads to **locally constant fit**
- slowly add in some points and let others gradually die off

\[
\hat{y}_q = \frac{\sum_{i=1}^{N} \text{Kernel}_\lambda(\text{distance}(x_i, x_q)) \cdot y_i}{\sum_{i=1}^{N} \text{Kernel}_\lambda(\text{distance}(x_i, x_q))}
\]

Local linear regression

So far, fitting **constant function locally** at each point
→ “locally weighted averages”

Can instead fit a **line or polynomial locally** at each point
→ “locally weighted linear regression”
Local regression rules of thumb

- Local linear fit reduces bias at boundaries with minimum increase in variance
- Local quadratic fit doesn't help at boundaries and increases variance, but does help capture curvature in the interior
- With sufficient data, local polynomials of odd degree dominate those of even degree

Recommended default choice: local linear regression

Discussion on k-NN and kernel regression
Nonparametric approaches

k-NN and kernel regression are examples of nonparametric regression.

General goals of nonparametrics:
- Flexibility
- Make few assumptions about \( f(x) \)
- Complexity can grow with the number of observations \( N \)

Lots of other choices:
- Splines, trees, locally weighted structured regression models...

Limiting behavior of NN: **Noiseless** setting (\( \epsilon_i=0 \))

In the limit of getting an infinite amount of noiseless data, the MSE of 1-NN fit goes to 0.
Limiting behavior of NN: **Noiseless** setting ($\varepsilon_i=0$)

In the limit of getting an infinite amount of noiseless data, the **MSE of 1-NN fit goes to 0**

![Graphs showing 1-NN fit and Quadratic fit with a comparison note](image)

Not true for parametric models!

---

**Error vs. amount of data**

For a fixed model complexity:

- Training error decreases with more data points.
- True error decreases with more data points.

In the limit of getting infinite amount of data, the true error of the model should be close to the true relationship in the noiseless setting.
Limiting behavior of NN: Noisy data setting

In the limit of getting an infinite amount of data, the MSE of NN fit goes to 0 if $k$ grows, too

NN and kernel methods for large $d$ or small $N$

NN and kernel methods work well when the data cover the space, but...
- the more dimensions $d$ you have, the more points $N$ you need to cover the space
- need $N = O(\exp(d))$ data points for good performance

This is where parametric models become useful...
Complexity of NN search

Naïve approach: Brute force search
- Given a query point $x_q$
- Scan through each point $x_1, x_2, \ldots, x_N$
- $O(N)$ distance computations per 1-NN query!
- $O(N \log k)$ per k-NN query!

What if $N$ is huge???
(and many queries)

KD-trees!
Locality-sensitive hashing, etc.

$k$-NN for classification
Spam filtering example

Input: $x$

Output: $y$

Text of email, sender, IP,...

Using k-NN for classification

Space of labeled emails (not spam vs. spam), organized by similarity of text

query email

not spam vs. spam: decide via majority vote of k-NN
Using k-NN for classification

Space of labeled emails (not spam vs. spam), organized by similarity of text

query email

not spam vs. spam

decide via majority vote of k-NN

Summary for nearest neighbor and kernel regression
What you can do now...

- Motivate the use of nearest neighbor (NN) regression
- Perform k-NN regression
- Analyze computational costs of these algorithms
- Discuss sensitivity of NN to lack of data, dimensionality, and noise
- Perform weighted k-NN and define weights using a kernel
- Define and implement kernel regression
- Describe the effect of varying the kernel bandwidth $\lambda$ or # of nearest neighbors $k$
- Select $\lambda$ or $k$ using cross validation
- Compare and contrast kernel regression with a global average fit
- Define what makes an approach nonparametric and why NN and kernel regression are considered nonparametric methods
- Analyze the limiting behavior of NN regression
- Use NN for classification

Clustering:
Grouping
Related Docs
Motivating clustering approaches

Goal: Structure documents by topic

Discover groups (*clusters*) of related articles

SPORTS

WORLD NEWS
Why might clustering be useful?

I don’t just like sports!

Learn user preferences

Set of clustered documents read by user

Cluster 1

Cluster 2

Cluster 3

Cluster 4

Use feedback to learn user preferences over topics
Clustering: An unsupervised learning task

What if some of the labels are known?
Training set of labeled docs
Multiclass classification problem

Example of supervised learning

Clustering

No labels provided
...uncover cluster structure from input alone

Input: docs as vectors $x_i$
Output: cluster labels $z_i$

An unsupervised learning task
What defines a cluster?

Cluster defined by center & shape/spread

Assign observation $x_i$ (doc) to cluster $k$ (topic label) if
- Score under cluster $k$ is higher than under others
- For simplicity, often define score as distance to cluster center (ignoring shape)

Hope for unsupervised learning

Easy

Impossible

In between
Other (challenging!) clusters to discover...

![Cluster Visualization](image)

Other (challenging!) clusters to discover...

![Cluster Visualization](image)
**k-means: A clustering algorithm**

Assume

- Score = distance to cluster center
  (smaller better)
k-means algorithm

0. Initialize cluster centers
   \( \mu_1, \mu_2, \ldots, \mu_k \)

1. Assign observations to closest cluster center

2. Revise cluster centers as mean of assigned observations

3. Repeat 1.+2. until convergence

\[
z_i \leftarrow \arg \min_j \| \mu_j - x_i \|_2^2
\]

Inferred label for obs \( i \), whereas supervised learning has given label \( y_i \).
k-means algorithm

0. Initialize cluster centers
1. Assign observations to closest cluster center
2. Revise cluster centers as mean of assigned observations
3. Repeat 1.+2. until convergence
Convergence of k-means

Converges to:

- Global optimum
- Local optimum
- Neither

Convergence of k-means to local mode
Convergence of k-means to local mode

- Convergence of k-means to local mode

  - Only diff (sometimes assigned to upper cluster, sometimes middle cluster)
Smart initialization with k-means++

k-means++ overview

Initialization of k-means algorithm is critical to quality of local optima found

Smart initialization:
1. Choose first cluster center uniformly at random from data points
2. For each obs \( x \), compute distance \( d(x) \) to nearest cluster center
3. Choose new cluster center from amongst data points, with probability of \( x \) being chosen proportional to \( d(x)^2 \)
4. Repeat Steps 2 and 3 until \( k \) centers have been chosen
k-means++ visualized

more likely to select a data point as a cluster center if it is far away from other cluster centers (\(\text{dist}^2\) increases this effect)
k-means++ visualized
k-means++ pros/cons

Computationally costly relative to random initialization, but the subsequent k-means often converges more rapidly.

Tends to improve quality of local optimum and lower runtime.

Assessing quality of the clustering and choosing the # of clusters.
Which clustering do I prefer?

---

**k-means objective**

$k$-means is trying to minimize the sum of squared distances:

$$\sum_{k} \sum_{j=1}^{k} \sum_{i: z_i = j} \| \mu_j - x_i \|^2$$

$$\Rightarrow \min_{\mathbf{\mu}_1, \ldots, \mathbf{\mu}_k} \text{sum over sq.dist. in cluster } j$$
Cluster heterogeneity

Measure of quality of given clustering:

\[ \sum_{j=1}^{k} \sum_{i: z_i = j} \| \mu_j - x_i \|^2 \]

Lower is better!

What happens as k increases?

Can refine clusters more and more to the data → overfitting!

**Extreme case** of k=N:
- can set each cluster center equal to datapoint
- heterogeneity = 0! (all distances to cluster centers are 0)

Lowest possible cluster heterogeneity decreases with increasing k
How to choose $k$?

- Lowest possible cluster heterogeneity
- # of clusters $k$

Other examples
Clustering images

• For search, group as:
  – Ocean
  – Pink flower
  – Dog
  – Sunset
  – Clouds
  – ...

Structuring web search results

• Search terms can have multiple meanings
• Example: “cardinal”

• Use clustering to structure output
Grouping patients by medical condition

• Better characterize subpopulations and diseases

Example: Patients and seizures are diverse
Cluster seizures by observed time courses

Products on Amazon

• Discover product categories from purchase histories

• Or discovering groups of users
Discovering similar neighborhoods

**Task 1:** Estimate price at a small regional level

**Challenge:**
- Only a few (or no!) sales in each region per month

**Solution:**
- Cluster regions with similar trends and share information within a cluster

![City of Seattle](image)

---

Discovering similar neighborhoods

**Task 2:** Forecast violent crimes to better task police

- Again, *cluster regions* and share information!
- Leads to *improved predictions* compared to examining each region independently

![Washington, DC](image)